

TFAE Grammar

```
<TFAE> ::= <num>
          | {+ <TFAE> <TFAE>}
          | {- <TFAE> <TFAE>}
          | <id>
          | {fun {<id> : <TE>} <TFAE>}
          | {<TFAE> <TFAE>}

<TE> ::= num
        | (<TE> -> <TE>)
```

TFAE Types

```
(define-type Type
  [numT]
  [arrowT (arg : Type)
           (result : Type)])
```

```
(define-type TypeEnv
  [mtEnv]
  [aBind (name : symbol)
         (type : Type)
         (rest : TypeEnv)])
```

TFAE Expressions

```
(define-type TFAE
  [num (n : number)]
  [add (l : TFAE)
       (r : TFAE)]
  [sub (l : TFAE)
       (r : TFAE)]
  [id (name : symbol)]
  [fun (name : symbol)
       (t : Type)
       (body : TFAE)]
  [app (rator : TFAE)
       (rand : TFAE)])
```

TFAE Type Checker

```
(define typecheck : (TFAE TypeEnv -> Type)
  (lambda (fae env)
    (type-case TFAE fae
      ...)))
```

TFAE Type Checker

```
(define typecheck : (TFAE TypeEnv -> Type)
  (lambda (fae env)
    (type-case TFAE fae
      [num (n) ...])))
```

$\Gamma \vdash \langle \text{num} \rangle : \text{num}$

TFAE Type Checker

```
(define typecheck : (TFAE TypeEnv -> Type)
  (lambda (fae env)
    (type-case TFAE fae
      [num (n) (numT)])))
```

$\Gamma \vdash \langle \text{num} \rangle : \text{num}$

TFAE Type Checker

```
(define typecheck : (TFAE TypeEnv -> Type)
  (lambda (fae env)
    (type-case TFAE fae
      [add (l r)
        ... (typecheck l env) ...
        ... (typecheck r env) ...])))
```

$$\frac{\Gamma \vdash \mathbf{e}_1 : \mathit{num} \quad \Gamma \vdash \mathbf{e}_2 : \mathit{num}}{\Gamma \vdash \{+ \mathbf{e}_1 \ \mathbf{e}_2\} : \mathit{num}}$$

TFAE Type Checker

```
(define typecheck : (TFAE TypeEnv -> Type)
  (lambda (fae env)
    (type-case TFAE fae
      [add (l r)
        (type-case Type (typecheck l env)
          [numT ()
            ... (typecheck r env) ...]
          [else (error 'typecheck
                       "add expects a num")]])))))
```

$$\frac{\Gamma \vdash \mathbf{e}_1 : \mathit{num} \quad \Gamma \vdash \mathbf{e}_2 : \mathit{num}}{\Gamma \vdash \{+ \mathbf{e}_1 \ \mathbf{e}_2\} : \mathit{num}}$$

TFAE Type Checker

```
(define typecheck : (TFAE TypeEnv -> Type)
  (lambda (fae env)
    (type-case TFAE fae
      [add (l r)
        (type-case Type (typecheck l env)
          [numT ()
            (type-case Type (typecheck r env)
              [numT () (numT)]
              [else (error 'typecheck
                "add expects a num")]])]
          [else (error 'typecheck
            "add expects a num")])])])])])])])
```

$$\frac{\Gamma \vdash \mathbf{e}_1 : \mathit{num} \quad \Gamma \vdash \mathbf{e}_2 : \mathit{num}}{\Gamma \vdash \{+ \mathbf{e}_1 \ \mathbf{e}_2\} : \mathit{num}}$$

TFAE Type Checker

```
(define typecheck : (TFAE TypeEnv -> Type)
  (lambda (fae env)
    (type-case TFAE fae
      [id (name) ...])))
```

$$[\dots \langle id \rangle \leftarrow \tau \dots] \vdash \langle id \rangle : \tau$$

TFAE Type Checker

```
(define typecheck : (TFAE TypeEnv -> Type)
  (lambda (fae env)
    (type-case TFAE fae
      [id (name) (get-type name env)])))
```

$$[\dots \langle id \rangle \leftarrow \tau \dots] \vdash \langle id \rangle : \tau$$

TFAE Type Checker

```
(define typecheck : (TFAE TypeEnv -> Type)
  (lambda (fae env)
    (type-case TFAE fae
      [fun (name arg-type body)
        ...])))
```

$$\frac{\Gamma[\langle \text{id} \rangle \leftarrow \tau_1] \vdash \mathbf{e} : \tau_2}{\Gamma \vdash \{\mathbf{fun} \{\langle \text{id} \rangle : \tau_1\} \mathbf{e}\} : (\tau_1 \rightarrow \tau_2)}$$

TFAE Type Checker

```
(define typecheck : (TFAE TypeEnv -> Type)
  (lambda (fae env)
    (type-case TFAE fae
      [fun (name arg-type body)
        ... (typecheck body (aBind name
                                   arg-type
                                   env))
        ...])))
```

$$\frac{\Gamma[\langle \text{id} \rangle \leftarrow \tau_1] \vdash \mathbf{e} : \tau_2}{\Gamma \vdash \{\text{fun } \{\langle \text{id} \rangle : \tau_1\} \mathbf{e}\} : (\tau_1 \rightarrow \tau_2)}$$

TFAE Type Checker

```
(define typecheck : (TFAE TypeEnv -> Type)
  (lambda (fae env)
    (type-case TFAE fae
      [fun (name arg-type body)
        (arrowT arg-type
          (typecheck body (aBind name
                                arg-type
                                env))))]))
```

$$\frac{\Gamma[\langle \text{id} \rangle \leftarrow \tau_1] \vdash \mathbf{e} : \tau_2}{\Gamma \vdash \{\text{fun } \{\langle \text{id} \rangle : \tau_1\} \mathbf{e}\} : (\tau_1 \rightarrow \tau_2)}$$

TFAE Type Checker

```
(define typecheck : (TFAE TypeEnv -> Type)
  (lambda (fae env)
    (type-case TFAE fae
      [app (fn arg)
          ...])))
```

$$\frac{\Gamma \vdash \mathbf{e}_1 : (\tau_2 \rightarrow \tau_3) \quad \Gamma \vdash \mathbf{e}_2 : \tau_2}{\Gamma \vdash \{\mathbf{e}_1 \ \mathbf{e}_2\} : \tau_3}$$

TFAE Type Checker

```
(define typecheck : (TFAE TypeEnv -> Type)
  (lambda (fae env)
    (type-case TFAE fae
      [app (fn arg)
        ... (typecheck fn env) ...
        ... (typecheck arg env) ...])))
```

$$\frac{\Gamma \vdash \mathbf{e}_1 : (\tau_2 \rightarrow \tau_3) \quad \Gamma \vdash \mathbf{e}_2 : \tau_2}{\Gamma \vdash \{\mathbf{e}_1 \ \mathbf{e}_2\} : \tau_3}$$

TFAE Type Checker

```
(define typecheck : (TFAE TypeEnv -> Type)
  (lambda (fae env)
    (type-case TFAE fae
      [app (fn arg)
        (type-case Type (typecheck fn env)
          [arrowT (arg-type result-type)
            ... (typecheck arg env) ...]
          [else (error 'typecheck
            "app expects a function")]])))))
```

$$\frac{\Gamma \vdash \mathbf{e}_1 : (\tau_2 \rightarrow \tau_3) \quad \Gamma \vdash \mathbf{e}_2 : \tau_2}{\Gamma \vdash \{\mathbf{e}_1 \ \mathbf{e}_2\} : \tau_3}$$

TFAE Type Checker

```
(define typecheck : (TFAE TypeEnv -> Type)
  (lambda (fae env)
    (type-case TFAE fae
      [app (fn arg)
        (type-case Type (typecheck fn env)
          [arrowT (arg-type result-type)
            (if (equal? arg-type
                        (typecheck arg env))
                result-type
                (error
                 'typecheck
                 "arg mismatch")))]
          [else (error 'typecheck
                       "app expects a function")]])))))
```

$$\frac{\Gamma \vdash \mathbf{e}_1 : (\tau_2 \rightarrow \tau_3) \quad \Gamma \vdash \mathbf{e}_2 : \tau_2}{\Gamma \vdash \{\mathbf{e}_1 \ \mathbf{e}_2\} : \tau_3}$$

TFAE Type Checker

```
(define get-type : (symbol TypeEnv -> Type)
  (lambda (name env)
    (type-case TypeEnv env
      [mtEnv () (error 'typecheck
                       "unbound identifier")]
      [aBind (name2 type rest)
              (if (equal? name name2)
                  type
                  (get-type name rest))])))
```

Pairs

```
{with {pair : (num -> (num -> (num -> num)))}
  {fun {x : num}
    {fun {y : num}
      {fun {s : num}
        {if0 s x y}}}}}}
{with {fst : ((num -> num) -> num)
  {fun {p : (num -> num)}
    {p 0}}}}
{with {snd : ((num -> num) -> num)
  {fun {p : (num -> num)}
    {p 1}}}}
{snd {{pair 1} 2}}}}}}
```

Pairs

```
{with {pair : (bool -> (bool -> (num -> bool)))
      {fun {x : bool}
        {fun {y : bool}
          {fun {s : num}
            {if0 s x y}}}}}}
{with {fst : ((num -> bool) -> bool)
      {fun {p : (num -> bool)}
        {p 0}}}
{with {snd : ((num -> bool) -> bool)
      {fun {p : (num -> bool)}
        {p 1}}}
{snd {{pair true} false}}}}}
```

Pairs

```
{with {pair : (num -> (bool -> (num -> ...)))}
  {fun {x : num}
    {fun {y : bool}
      {fun {s : num}
        {if0 s x y}}}}}}
{with {fst : ((num -> ...) -> ...)}
  {fun {p : (num -> ...)}
    {p 0}}}}
{with {snd : ((num -> ...) -> ...)}
  {fun {p : (num -> ...)}
    {p 1}}}}
{snd {{pair 1} false}}}}}}
```

Pairs

```
{with {pair : (num -> (bool -> (num -> ...)))}
  {fun {x : num}
    {fun {y : bool}
      {fun {s : num}
        {if0 s x y}}}}}}
{with {fst : ((num -> ...) -> ...)}
  {fun {p : (num -> ...)}
    {p 0}}}}
{with {snd : ((num -> ...) -> ...)}
  {fun {p : (num -> ...)}
    {p 1}}}}
{snd {{pair 1} false}}}}}}
```

No possible type for ...

TPFAE Grammar

```
<TPFAE> ::= <num>
          | {+ <TPFAE> <TPFAE>}
          | {- <TPFAE> <TPFAE>}
          | <id>
          | {fun {<id> : <TE>} <TPFAE>}
          | {<TPFAE> <TPFAE>}
          | {pair <TPFAE> <TPFAE>}
          | {fst <TPFAE>}
          | {snd <TPFAE>}
```

NEW

NEW

NEW

```
<TE> ::= num
       | bool
       | (<TE> -> <TE>)
       | (<TE> * <TE>)
```

NEW

TPFAE Grammar

```
<TPFAE> ::= <num>
          | {+ <TPFAE> <TPFAE>}
          | {- <TPFAE> <TPFAE>}
          | <id>
          | {fun {<id> : <TE>} <TPFAE>}
          | {<TPFAE> <TPFAE>}
          | {pair <TPFAE> <TPFAE>}
          | {fst <TPFAE>}
          | {snd <TPFAE>}
```

NEW

NEW

NEW




```
<TE> ::= num
       | bool
       | (<TE> -> <TE>)
       | (<TE> * <TE>)
```


NEW

$$\Gamma \vdash \mathbf{e}_1 : \tau_1 \quad \Gamma \vdash \mathbf{e}_2 : \tau_2$$

$$\Gamma \vdash \{\mathbf{pair} \ \mathbf{e}_1 \ \mathbf{e}_2\} : (\tau_1 \times \tau_2)$$

TPFAE Grammar




<TPFAE> ::= **<num>**
| **{+ <TPFAE> <TPFAE>}**
| **{- <TPFAE> <TPFAE>}**
| **<id>**
| **{fun {<id> : <TE>} <TPFAE>}**
| **{<TPFAE> <TPFAE>}**
| **{pair <TPFAE> <TPFAE>}** 
| **{fst <TPFAE>}** 
| **{snd <TPFAE>}** 


<TE> ::= **num**
| **bool**
| **(<TE> -> <TE>)**
| **(<TE> * <TE>)** 

$$\Gamma \vdash \mathbf{e} : (\tau_1 \times \tau_2)$$

$$\Gamma \vdash \{\mathbf{fst} \ \mathbf{e}\} : \tau_1$$

TPFAE Grammar

<TPFAE> ::= **<num>**
| **{+ <TPFAE> <TPFAE>}**
| **{- <TPFAE> <TPFAE>}**
| **<id>**
| **{fun {<id> : <TE>} <TPFAE>}**
| **{<TPFAE> <TPFAE>}**
| **{pair <TPFAE> <TPFAE>}** 
| **{fst <TPFAE>}** 
| **{snd <TPFAE>}** 

<TE> ::= **num**
| **bool**
| **(<TE> -> <TE>)**
| **(<TE> * <TE>)** 

$$\Gamma \vdash \mathbf{e} : (\tau_1 \times \tau_2)$$

$$\Gamma \vdash \{\mathbf{snd} \mathbf{e}\} : \tau_2$$